the entry's miss and access counters 320, 330, respectively, are incremented. These counters 320, 330 saturate at their maximum value. If the set is not in the SMT 300 and the set experiences a hit, no action occurs.

If the set is not in the SMT 300 and a miss is experienced, the set's index is recorded in an invalid SMT entry if there is one. Entries may invalidate themselves under specified conditions. In this example, an entry invalidates itself if three or fewer misses occur over the last eight or more accesses. The logic 350 for this self-invalidation is shown in FIG. 3. If there are no invalid entries in the SMT 300, the least recently accessed entry is used for the new entry and its previous contents are overwritten.

If, after an access, an entry finds that it has had eight or more misses in the last eight or more accesses, a thrashed set signal is activated using logic 360. In this case, its hit rate is, at most, 50% and may be less. The thrashed set signal initiates the augmentation of the set with one or more other sets. The logic 360 for this approach is illustrated in FIG. 3.

Selection and Acquisition Of Augmentation Sets

To decrease thrashing on a set, one or more additional sets in a cache may be selected dynamically or assigned statically to share their space with the thrashed set. These sets may be selected, e.g., on the basis of a low access rate, their position in the address space relative to a thrashed set or their miss rate, using a static assignment, a wired-in assignment or by other means. When a given set augments a thrashed set, it shares its space with the thrashed set. In a unidirectional augmentation approach, some blocks that formerly mapped to the thrashed set now map to an augmentation set and blocks that previously mapped to the augmentation set continue to do so. In a bidirectional augmentation approach, the blocks that map to either the thrashed set or to the augmentation set are distributed across both sets. That is, some blocks that previously mapped to the thrashed set now map to the augmentation set and some blocks that previously mapped to the augmentation set now map to the thrashed set.

In an exemplary implementation of a mechanism to select the augmentation set(s), selection is based on a set's cache index relative to that of a thrashed set. In this approach, a thrashed set is augmented with one set (the augmentation set) when appropriate. The additional set is identified by its index relative to that of the thrashed set. The index of the selected augmentation set is that of the thrashed set with the most significant index bit inverted. For

example, if the index of the thrashed set is a binary value of "1111111," the index of the corresponding augmentation set is "0111111." The advantage of this approach is that it is static (and may be wired in) and is relatively easy to implement. Of course, the disadvantage is that the statically assigned augmentation set may not be the best choice of sets to serve this purpose. Decreased effectiveness is traded for increased implementation simplicity. This approach is referred to as the "static-pairs" approach with a set and its potential augmentation set comprising each pair.

At least two versions of the static-pairs approach are possible. In one variation, each set in the pair may act as an augmentation set for the other. In another variation, one set (A) in a pair may augment the other set (B), but set B may not augment set A. The first approach is referred to as symmetric static-pairs, and the second approach is referred to as asymmetric static-pairs.

An improved, albeit more complex, approach is one in which a group of sets are statically assigned to each set, to act as a pool of sets from which an augmentation set is selected. This is approach is referred to as the "static-group" approach. The set with the lowest access rate or lowest miss rate in the static group may be selected as the augmentation set. Other, dynamic approaches may be envisioned, such as dynamic-pairs and dynamic-group approaches, in which an augmentation set is selected from a group of sets whose membership is dynamic.

Mapping Thrashed Blocks To Augmentation Sets

There are many ways to map some of the blocks mapped to a thrashed set to an augmentation set (to couple the augmentation set to the thrashed set). Approaches are strongly influenced by the method used to select augmentation sets. Coupling should insure that blocks that are thrashing would be mapped in equal numbers to the thrashed set and to the augmentation set. However, this approach requires extensive logic that may involve recording the addresses of the blocks that are thrashing. A simpler approach is presented below.

The coupling approach presented is referred to as the concurrent symmetric static-pairs approach because each set in a pair is the augmentation set for the other and if thrashing is detected on either set, both sets share blocks mapped to them with the other set. In non-concurrent versions, the thrashed set shares blocks mapped to it with an augmentation set but the

augmentation set would not share its blocks with the thrashed set. An implementation of the concurrent approach is shown in FIG. 4.

FIG. 4 is a schematic block diagram of an exemplary implementation of a concurrent symmetric static-pairs coupling scheme 400. The exemplary cache 410 has 128 sets, or 64 set-pairs. The mechanism 400 has a 64-bit set-pair vector 420 that contains a bit for each set-pair. A bit in this vector 420 is selected with a set-pair index that is derived from a set index 430 as shown in FIG. 4. A binary value of "one" in the set-pair vector 420 specifies that the corresponding set-pair is thrashed and a binary value of "zero" specifies that the set-pair is not thrashed. An appropriate bit is set in this vector 420 when thrashing is detected (a "1") and when augmentation is no longer desired (a "0").

Suppose that the cache 410 is accessed with a block address 440 and that its set-pair index 430 selects a true bit in the set-pair vector 420. In this case, a low order bit in the block address 440 (a4 in this implementation) is substituted for the high order bit of the set index 430. This causes some blocks that previously mapped to one of the sets in the pair to be mapped to the other set. This very simple logic decreases the thrashing in the set, improving the hit rate.

A cache directory 450 is accessed with the modified index, but a block's unmodified address is always held in the directory 450. During an access, a block's unmodified address is compared with that held in the directory 450 as shown in FIG. 4. In this way, correct data is accessed before, during and after changes in the mapping of blocks to sets, with the exception of write back caches.

Write-Back Caches

The above approach works in instruction caches and write-through caches. If however, a set may contain data that has been written into it and is not reflected in main memory, a problem arises when a map is changed. An access involving a block that has been altered in the cache may miss using an altered map, resulting in incorrect data being retrieved from main memory. Therefore, if the present invention is employed in a write-back cache, either a set that contains an altered block (dirty block) may not be remapped, or the dirty block that it contains must be written back to main memory before the map is altered.